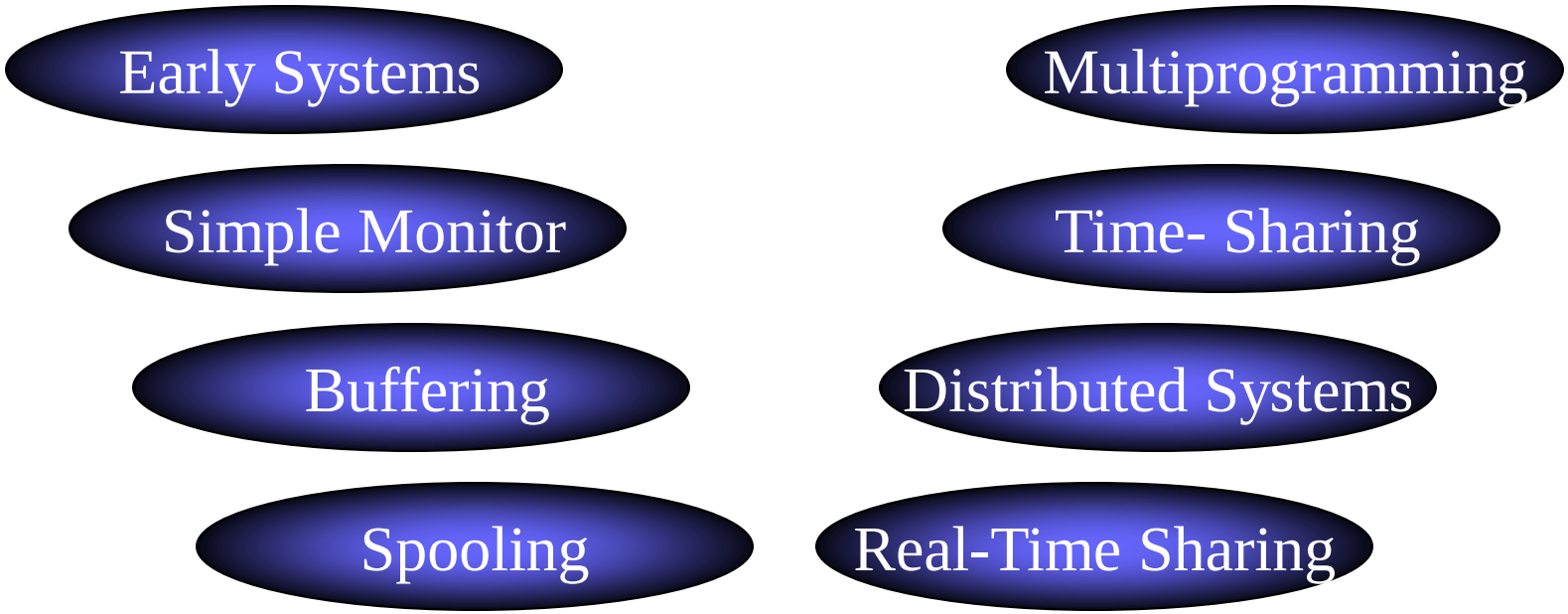
History of Operating System



History of Operating System

Early computers were physically very large machines running from a console (Figure 2.3) . The programmers would write a program and then operate the program directly from the operator's console. The program would be manually loaded into memory from the front panel switches ,paper tape or from punched cards. Then the appropriate buttons will be pushed to set the starting address and to start execution of the program. As the program run, the programmer/operator could monitor its execution by the display lights on the console. If errors were discovered, the programmer would

Early System

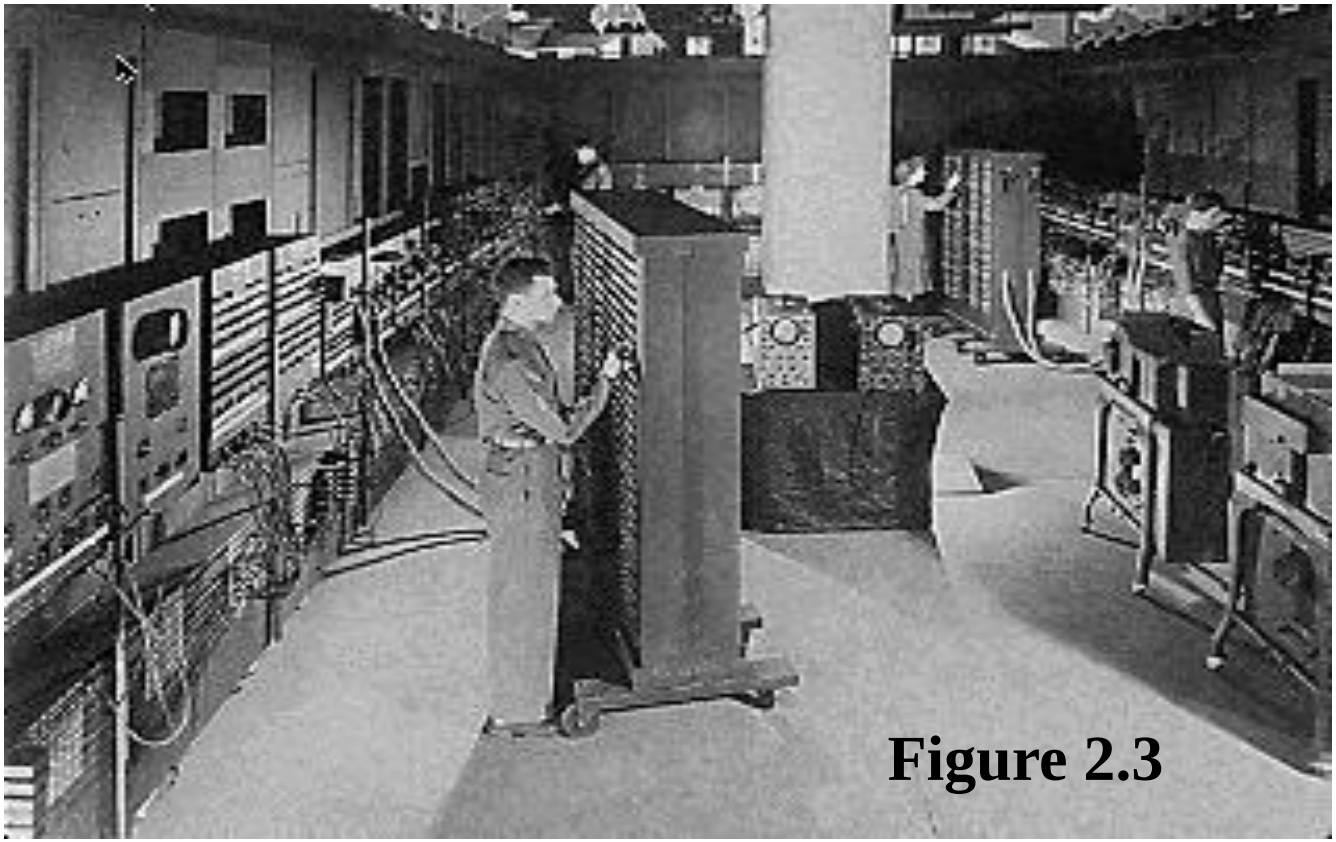


Figure 2.3

Early System

halt the program, examine the contents of memory and registers and debug the program directly from the console. Output was printed or was punched onto paper tape or cards for later printing.As time went on, additional software and hardware were developed. Card readers, line printers and magnetic tape became common. Assemblers, loaders and linkers were designed to ease the programming task. Libraries of common function were created. Common functions could then be copied into a new program without having to be written again.[SILBERSCHATZ,1994]

Early System

The routines that performed I/O were important. Each new I/O device has its own characteristics, requiring careful programming.. A special subroutine was written for each device. And such device is known as the *device driver.* [SILBERSCHATZ,1994]

Device Driver

DEVICE DRIVER of OS include system I/O buffers for each I/O device. A device driver knows how the buffers, flags, registers, control bits and status bits for each particular device should be used. Each device has its own driver, for example, a simple disk such as reading a character from a paper tape reader, might involve complex sequences of device-specific operations. Rather than the necessary code being written every time, the device driver was simple used from the library.[SILBERSCHATZ,1994}

Simple Monitor

During the early days, very few computers were available and they costs millions of dollars. That's why the owners of these computers wanted to get as much out of the machines.

The solution is two-fold:

1. A professional computer operator is hired. The programmer no longer operated the computer, instead they only provide the operator whatever cards or tapes needed and a short description of how to run the program.

2. The use of batch system of processing jobs. Jobs with similar needs were batched together and run through the computer as a group.

Simple Monitor

Continuation of...

Resident Monitor or simple monitor was created to automatically transfer control from one job to another. It is always resident in the memory.

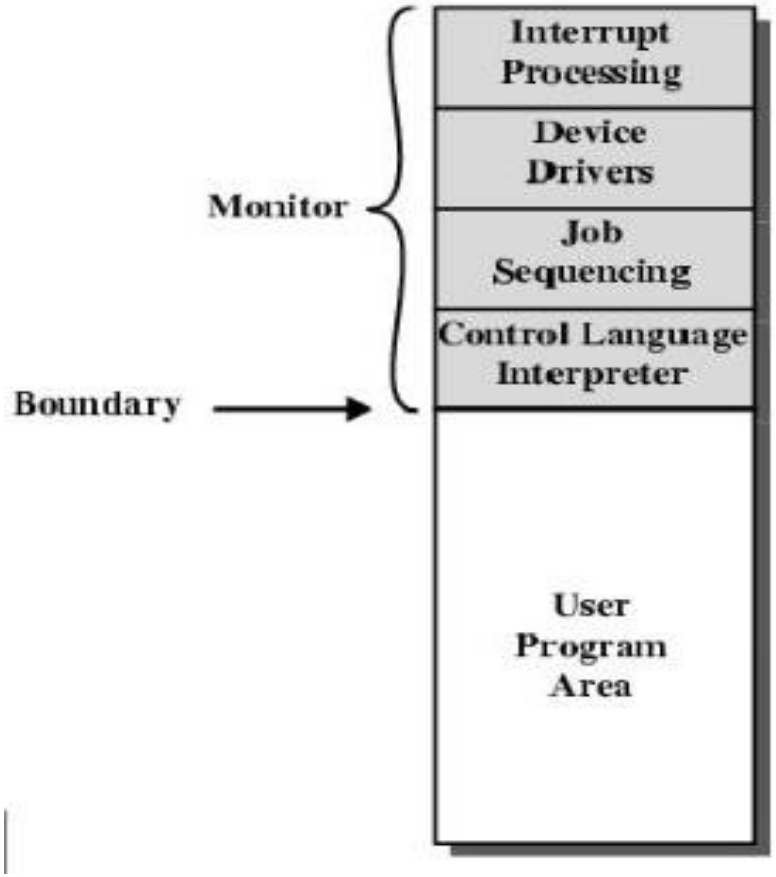
How Resident Monitor Works?

Initially, when the computer was turned on, the resident monitor was invoked and it would transfer control to a program. When the program terminated, it will return control to the resident monitor, which would then go to the next program. Thus, the resident monitor would automatically sequence from one program to another and from one job to another,

Parts of a Resident Monitor

- The control-card interpreter is responsible for reading and carrying out instructions on the cards at the point of execution.
- The loader to load systems programs and application programs into memory.

Memory layout for a resident monitor



Source: https://www.onlinenotesnepal.com/resident-monitor

The Resident monitors are divided into 4 parts as:

1. Control Language Interpreter:

The first part of the Resident monitor is a control language interpreter which is used to read and carry out the instruction from one level to the next level.

2. Loader:

The second part of the Resident monitor which is the main part of the Resident Monitor is the Loader which Loads all the necessary system and application programs into the main memory.

3. Device Driver:

This is used to managing the connecting input-output devices to the system. It is the interface between the user and the system. It works as an interface between the request and response to the request which user made.

4. Interrupt Processing:

The fourth part as the name suggests, processes the all occurred interrupt to the system.

Spooling

The word spooling is an acronym for **simultaneous peripheral operation on-line.** Spooling uses the disk (refer to the figure) as a very large buffer as far ahead as possible on input devices and storing output files until the output devices are able to accept them.

Spooling is also used for processing data at remote sites. The CPU sends the data via communications paths to a remote printer. The remote processing is done at its own speed without CPU intervention. The CPU just needs to be notified when the processing is completed, so that it can spool the next batch of data.[SILBERSCHATZ,1994]

Spooling

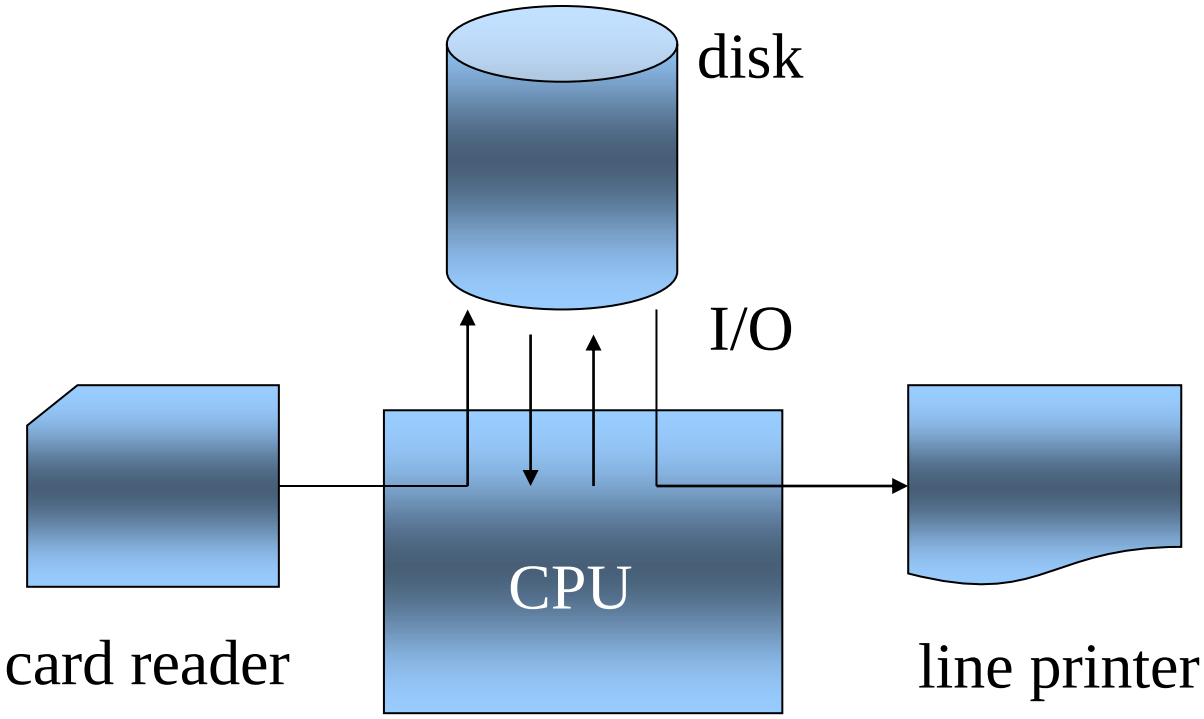


Fig.1.3

Spooling

During the idle periods, the operating system's spooling module reads data from such slow devices as card readers and terminal keyboards and stores them on a high-speed medium such as disk, even before the program needing those data has been loaded into memory. Later, when the program is loaded , its input data can be read from the disk. On output, the data are spooled to disk and later dumped to the printer. Thus, the application finishes processing and frees space for another program more quickly. [DAVIS,1992]

How Spooling Works?

1) Jobs comes as input arrive sequentially but stored temporarily on a disk.
2) A scheduler then selected the jobs from the pool in any desired order (prioritize).
3) Job being pulled from printing could be printed conveniently in any order.

Advantages of Spooling

- Spooling can keep both CPU and the I/O devices working at much higher rates, particularly if there is a mix of CPU bound and I/O bound jobs to be run.
- Spooling also provide a very important data structure- a job pool.

Components of Spooling

I. Writer mechanism

II. Reader mechanism

III. Scheduler

What is a Job Pool?

Job Pool is several jobs on disk that have already been read and waiting on disk, ready to be run.Spooling will greatly result in several jobs that had already been read waiting on disk, ready to run. A pool of jobs on disk allows the operating system to select which job to run next, in order to increase CPU utilization.

Buffer

Buffer is an area of memory used for holding data during input/output transfers to and from disk or tape, from the keyboard, to the printer and so on. For example, when characters are typed at a keyboard, they are placed in a buffer by an I/O channel, when the transfer is complete (i.e. the user presses the <enter> key) they may be accessed by the CPU is free. Similarly, information that is to be printed is placed in a print buffer which may be up to a megabyte long, located either in the printer or the

Buffer

Continuation of...

computer, or both. The CPU then issues the ‘start I/O’ instruction and the I/O channel transfers the data from the buffer to the printer. This process of being able to perform input and output independently of the CPU is known as autonomous operation of peripherals.

Buffering

BUFFERING – is a method of overlapping the I/O of a job with its own computation. Buffering is generally an operating-system function. Buffering mainly helps to smooth over variation in the time it takes to process a record.

Hardware Buffer

Hardware Buffer – a storage unit of an I/O device or associated control unit, used for temporary storage of data to be transferred between the device and the CPU or the memory.

Buffering

To solve the problem of notification, 2 ways were developed to handle this:

1) Polling – the querying of all I/O devices to detect which request service. Scanning the networks of terminals of terminals or sensors by the computer and asking one after the other if there is any data to be transmitted. Polling require the CPU to check regularly a channel to see if servicing was needed.

Buffering

2)Hardware Interrupts – seemed better than polling. It allows the channel to interrupt the processor when service is required.

-a signal that can't be ignored for every long period of time

-a signal that causes the processor to shift gears and undertake some new specific action.

Buffering Process

1) Data have been read, CPU will operate this but will instruct device to begin next input.

2) By the time that the CPU is ready for the next data (record), the input device will have finished reading it.

3) Repeat process again.
Similar process is done for output. CPU creates data that are put into a buffer until an output device can accept them.

Buffering Conditions

1) If CPU finishes first, it must wait for the next record to be processed.

2) If input finish first, either it must wait or it may proceed with reading the next record.

3) If input is faster than CPU, buffer will become full and input devices will have to wait.

Advantages of Buffering

• If the CPU (average speed DEVICE DRIVER or the resident monitor of OS include system I/O buffers for each I/O device.

• If the CPU is always faster, then it will always find an empty buffer and have to wait for the input device.

Advantages of Buffering

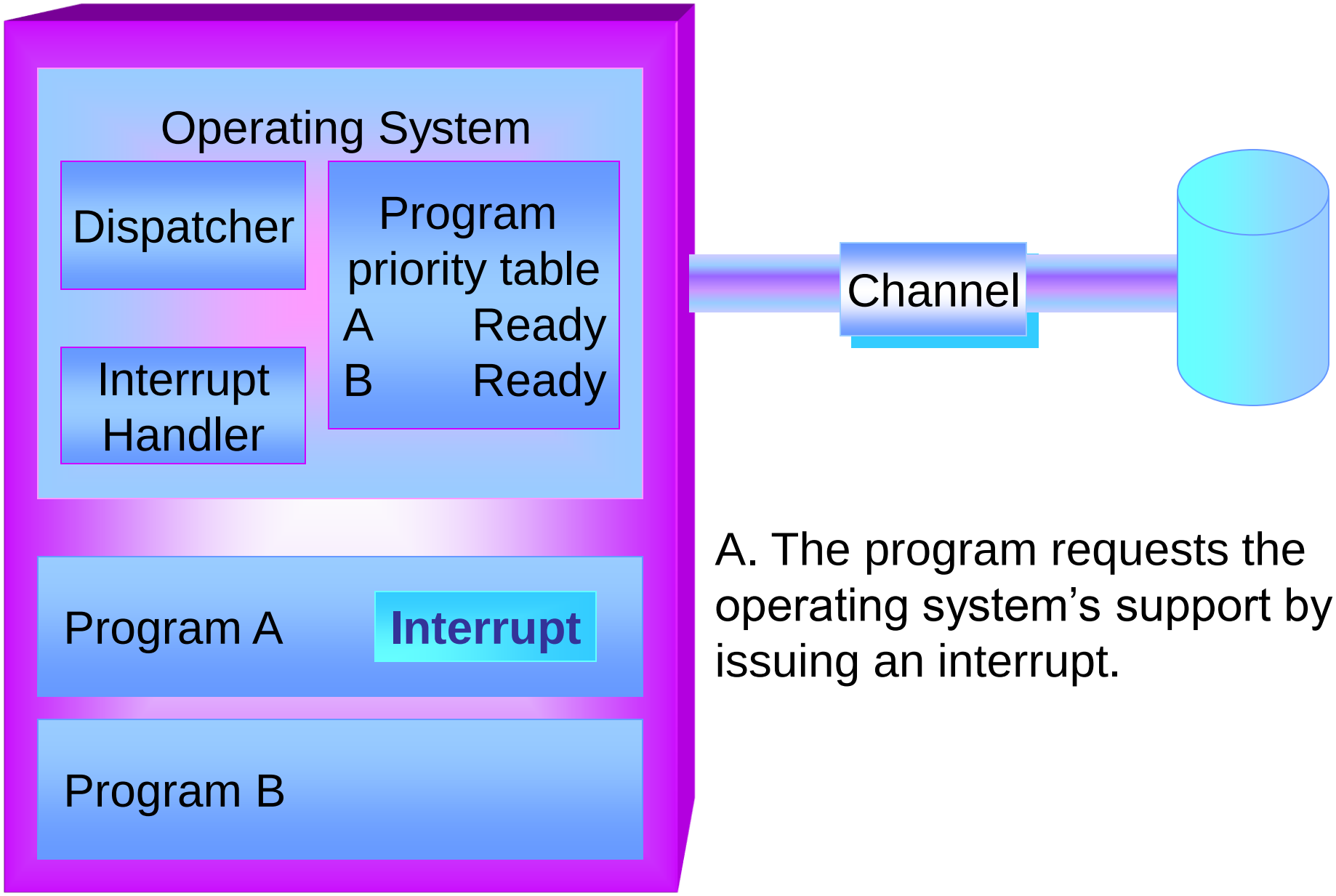
- The speed of execution is bounded by the speed of the input/output device not by the speed of CPU since CPU is faster than input/output.
- Card readers, line printers, paper tape readers and punchers were simply too slow to keep up with the CPU.

Single Buffering

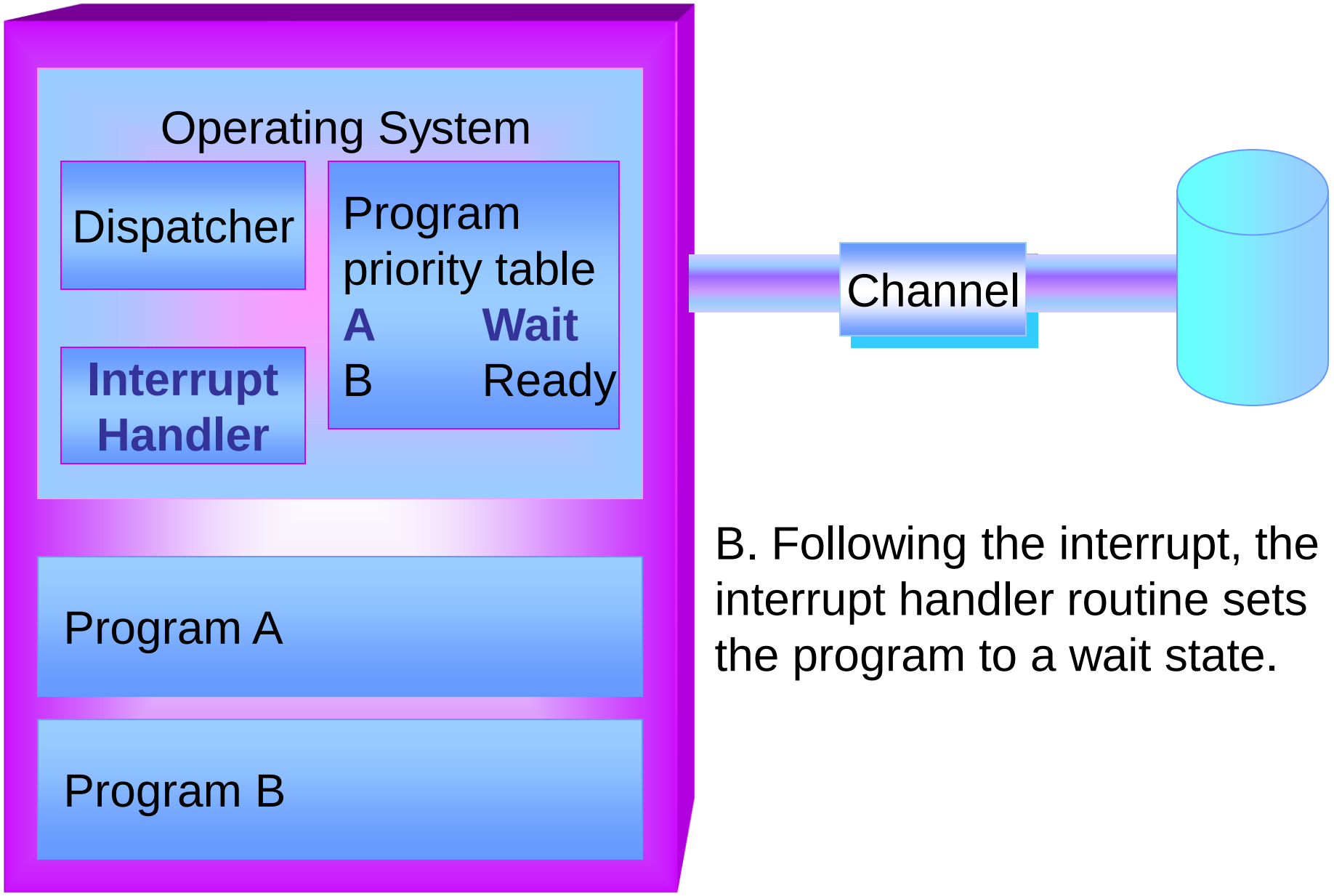
With single-buffered input, the input channel deposits data in the buffer, the processor processes it; more data is deposited, and so on. While the channel is depositing data, no processing can occur, and while the data is being processed, no data can be deposited.

Interrupts

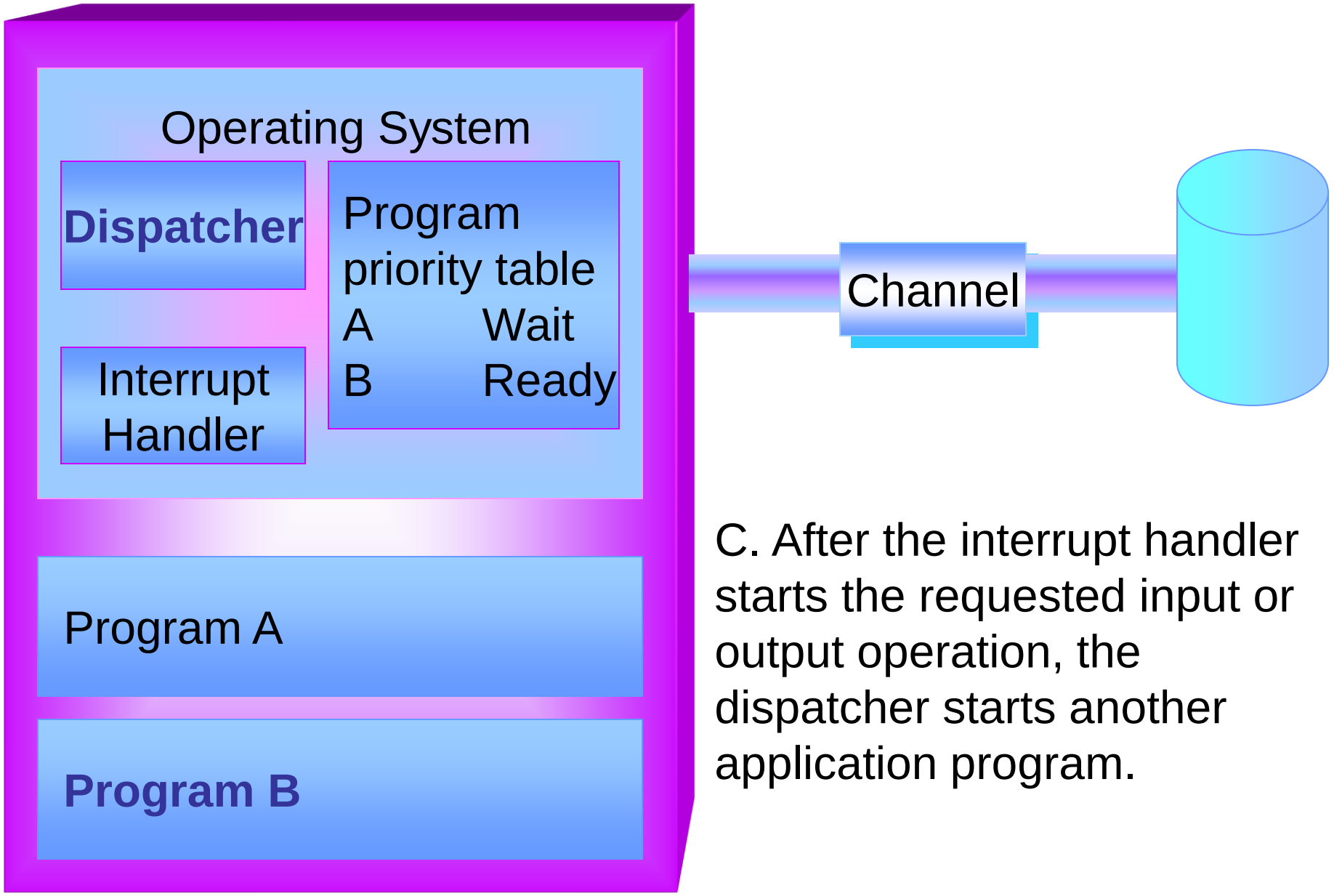
An interrupt is an electronic signal. Hardware senses the signal, saves the key control information fort he currently executing program, and transfers control to the operating system’s interrupt handler routine.At that instant, the interrupt ends. The operating system then handles the interrupt, and the dispatcher, subsequently, starts an application program. Eventually, the program that was executing at the time of the interrupt resumes processing.



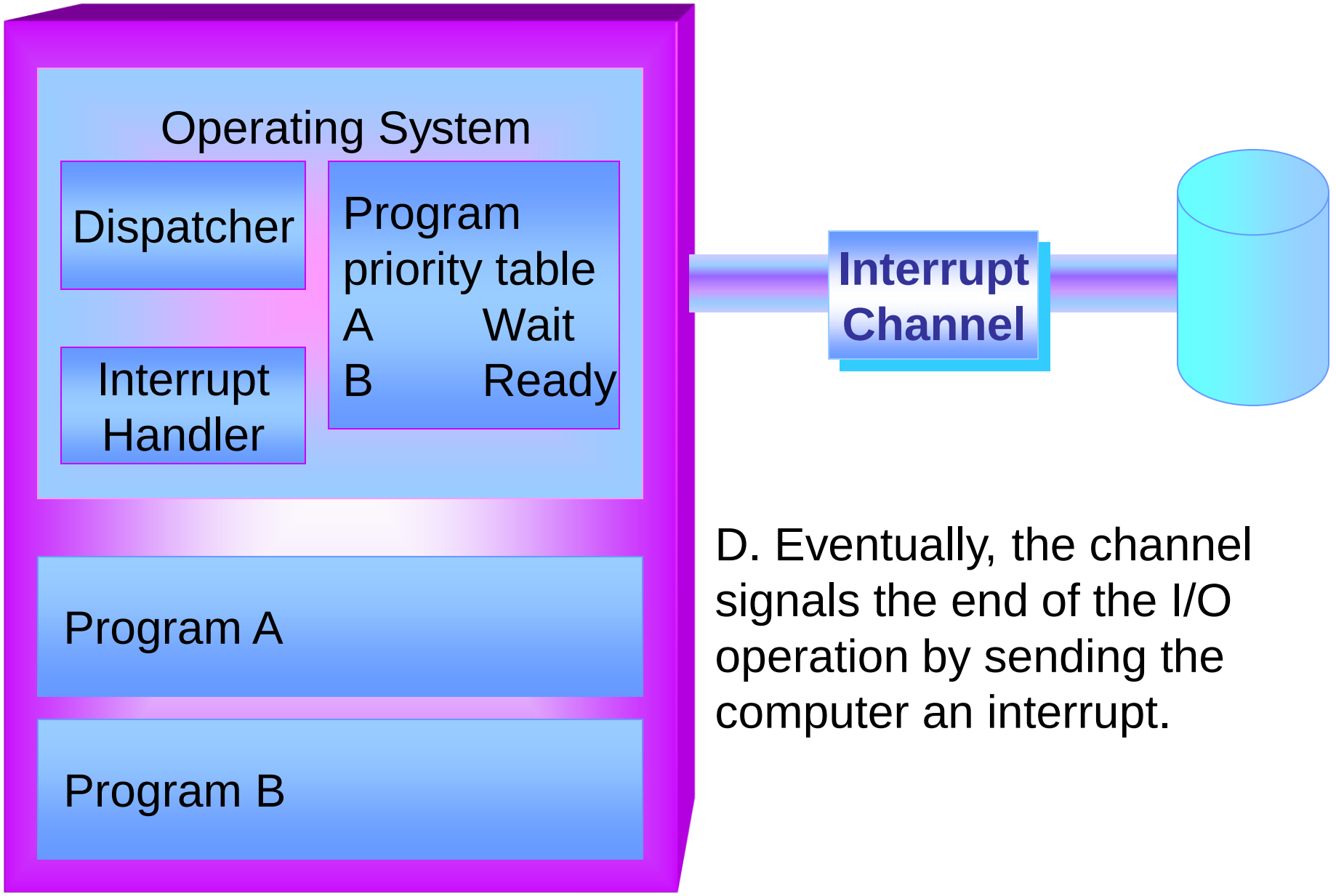
A. The program requests the operating system's support by issuing an interrupt.



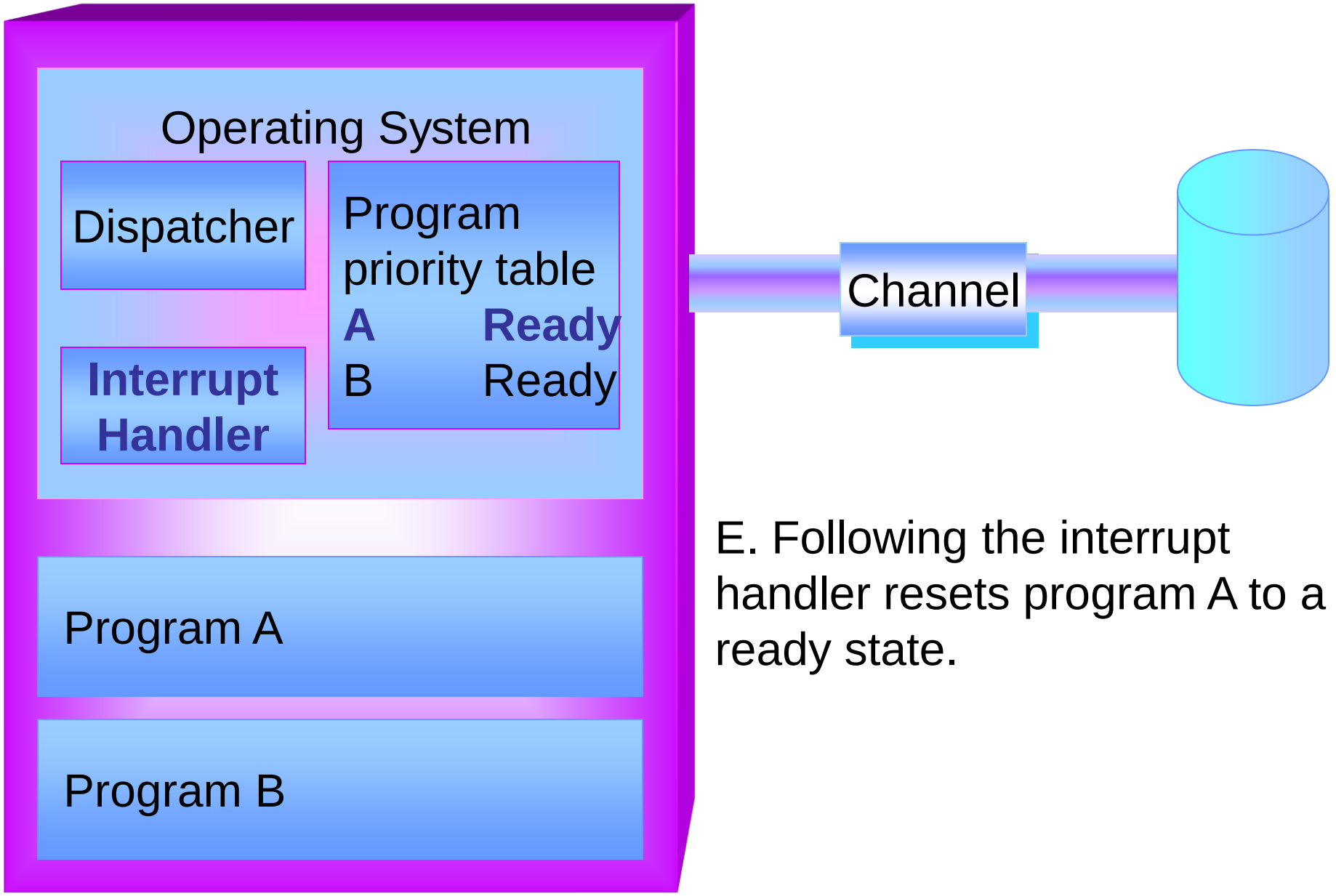
B. Following the interrupt, the interrupt handler routine sets the program to a wait state.



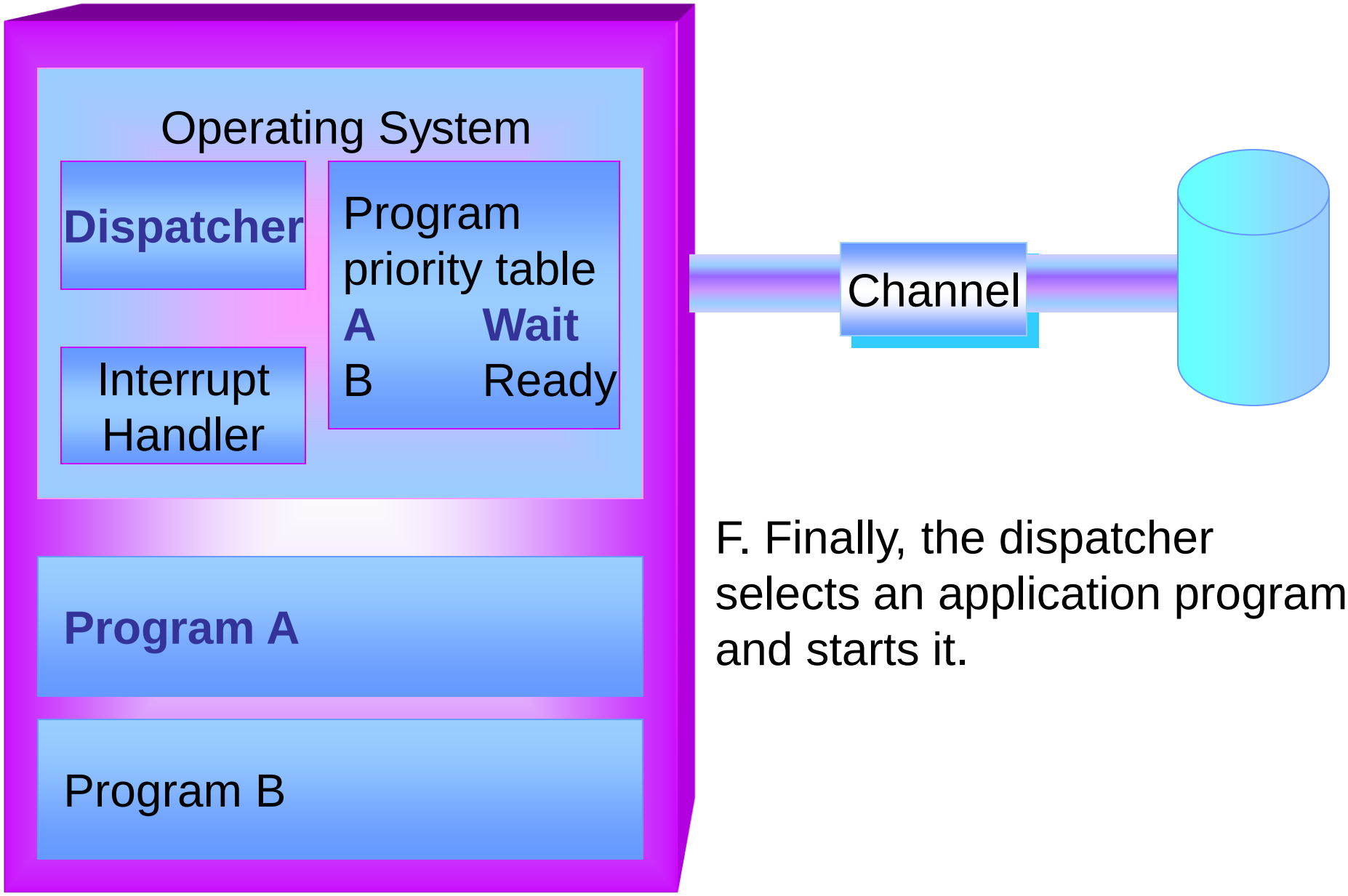
C. After the interrupt handler starts the requested input or output operation, the dispatcher starts another application program.



D. Eventually, the channel signals the end of the I/O operation by sending the computer an interrupt.



E. Following the interrupt handler resets program A to a ready state.



F. Finally, the dispatcher selects an application program and starts it.

Interrupts

The address of the next instruction is kept in a register called the PSW (program status word) and this also indicates the types of interrupts currently enabled and those currently disabled interrupts remain pending, or in some cases are ignored.

In smaller systems, the Operating System handles all interrupts by itself, which means that the interrupts are disabled for larger proportion of time.

How Interrupt Works?

HOW THE INTERRUPT MECHANISM WORKS?

There is a special register in the CPU called the INTERRUPT REGISTER. At the beginning of each fetch-execute cycle, the interrupt register is checked. Each bit of the register represents a different type of interrupt, and a bit is set, the state of the current process is saved and the Operating System routes control to the appropriate interrupt handler.

Since more than one device may request an interrupt simultaneously, each device is assigned a PRIORITY. Low

How Interrupt Works?

speed devices such as terminals and printers are given a high priority, since they are more liable to left behind with what they are doing, and so should be allowed to start as soon as possible so that they do eventually hold up processing.

In some cases if an interrupt occurs during data transfer, some data could be lost, and so the Operating System will disable user interrupts until it completes its task.

In large multi-user system there is a constant

How Interrupt Works?

stream of interrupts directed at the processor, and it must respond quickly as possible to these in order to provide an acceptable response time. Once an interrupt is received, the OS disables interrupts while it deals with the current interrupt. Since this could mean that interrupts are disabled for a large proportion of time, the nucleus (i.e. the part of OS that is always at the main store) on large systems simply determines the cause of the interrupt and then passes the problem to the specific interrupt handler, leaving itself free to deal with the next interrupt.

Example Illustration of How Interrupt Works

Program A is the currently running process. It needs to retrieve some data from disk, so an interrupt is generated. The interrupt handler changes the status of A to “blocked”, makes a request to the disk drive for data, and invokes a program called the dispatcher which selects Job B to run next. After a while, the disk drive has filled the buffer area and generates an interrupt to say it is ready. The interrupt handler is invoked, and changes the status of program A from ‘blocked’ to ‘runnable’, but B is left running. One millisecond later B’s time up is called by the interrupting clock and the dispatcher hands the CPU back to A, leaving B’s status as ‘runnable’.

Types of Interrupts

- Program check interrupts – caused by various types of error such as division by zero.
- Machine check interrupts – caused by malfunctioning hardware.

Example of Interrupts

- Int 20H

  Program terminate; use to exit from the program and pass control to DOS; don’t close automatically the file open w/ 2111

- Int 21H

  General DOS Service; it can take advantage of wide range of DOS function through 21H

Multiprogramming

A multiprogramming operating system begins with the same basic functions as a single -user operating system. Generally, key tables and control blocks occupy low memory. They are followed by input/output control system, a file system, and a command processor. Next comes logic to manage the system's resources, including memory management and memory protection routines, a dispatcher, one or more interrupt handler routines, device allocation modules, logic to deal with or prevent deadlocks, a queuing routine, a scheduler and a spooler. [DAVIS,1992]

Time-Sharing

Time Sharing (or multitasking) is a logical extension of multiprogramming. The idea of time sharing was demonstrated as early as 1960, but since time-shared systems are more expensive and difficult to built, (due to the numerous I/O devices needed), they did not become common until the early 1970's. Time-sharing systems were developed to provide interactive use of a computer system at a reasonable cost. A time-shared operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer. Each user has a separate program in the memory. A time-shared operating system allows the many users to share the computer simultaneously. Since action or command in a

Time-Sharing

Continuation of...

time-shared system tends to be short,only a little CPU time is needed for each user. As the system switches rapidly from user to the next, each user is given the impression that he has his own computer , whereas actually one computer is being shared among many users.

Characteristics of a Time-Sharing System

- Time sharing operating system are sophisticated.
- They provide a mechanism for concurrent execution.
- They also provide disk management.
- They also provide an on-line file system.

Time-Sharing Techniques for Memory Management

- Roll-in/Roll-out

Most time-sharing systems use such roll-in/roll-out (or swap-in/swap-out) techniques to manage memory space.

- Time-Slicing

In time-slicing, each program is restricted to a maximum “slice” of time, perhaps 0.01 second. Once a program gets control, it runs until one of two things happens. If the program requires input or output before

Time-Sharing Techniques for Memory Management

exhausting is time slice, it calls the operating system and "voluntarily" drops into a wait state. If the program uses up its entire time slice, a timer interrupt transfers control to the operating system which selects the next program.

- Polling

A polling algorithm is used to determine which programs is activated next.

Real-Time Sharing

Real time sharing is a system which has a capability of a fast response to obtain data from an activity on a physical process.

Characterized by processing activity triggered by external events. The processing of each task must be completed within a rigid time constraints. Responsiveness to the environment is more important than the utilization of resources. Usually used to monitor production lines, control continuous processes, etc.

Multi-Processing System

Most systems are single-processor systems, that is, they have only one main CPU. However, there is an increasing trend toward multiprocessor system. Such systems have more than one CPU in close communication, sharing the computer bus and sometimes memory and peripheral devices.

Requirements of a Multi-Processing System

- Inter-processor communication
- More complicated process scheduling
- Parallel processing synchronization

Advantages of a Multi-Processing System

1. Throughput- increasing the number of processors can get more work done in a shorter period of time.

2. Reliability- even if one processor fails, all other processors remain in function and continue process.

3. File System Manipulation- It should be obvious that programs need to read and write files as they also need to create and delete files by name.

4. Communication- Computation maybe implemented via shared memory, or by message passing in which packets of information are moved between processes by the operating system.

Distributed Systems

Distributed systems refers to the computer systems that distributes computation among several physical processors. The processors may vary in size and function.

Schemes of a Distributed Systems

- Tightly coupled system

  ⇒Sharing the computer bus, the clock, memory and peripherals
  ⇒Single Operating System

- Loosely coupled

  ⇒Separate memory, Operating System connected via communication link or bus

  ⇒Total transparency to user

Advantages of Distributed Systems

- Better price/performance than mainframes
- If one machine crashes, the system as a whole can still survive
- Computing power can be added in small increments

Disadvantages of Distributed Systems

- Little commercial software exists as present
- Network may cause problems.
- Easy access and subjected to secure threats and attack